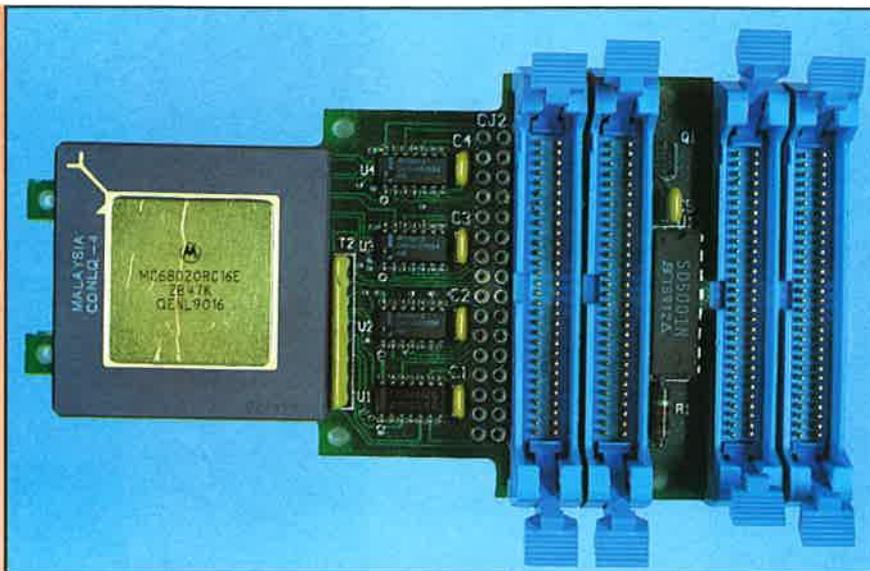
EMULATORI DI MICROPROCESSORI

Gli emulatori di microprocessori rappresentano lo strumento più potente attualmente disponibile in commercio per la progettazione hardware e software dei sistemi a microprocessore.

I metodi tradizionali di controllo e regolazione delle apparecchiature a microprocessore hanno subito una notevole evoluzione grazie alla comparsa di questo tipo di strumenti. Con questi dispositivi l'individuazione di un errore presente nel programma del microcontroller non rappresenta più la "ricerca di un ago nel pagliaio".

Un emulatore sostituisce il microprocessore di un sistema sotto prova, comportandosi allo stesso modo

L'EMULATORE

L'emulatore è in grado di sostituire completamente il microprocessore del sistema che si sta analizzando, funzionando esattamente allo stesso modo ma permettendo il monitoraggio simultaneo delle operazioni eseguite e la

Pinza di prova per il microprocessore 68020 della Motorola

loro alterazione in caso di necessità. Ad esempio, è possibile indicare all'emulatore di eseguire degli stop *(breakpoint)* in punti determinati del programma; finché non si raggiungono questi punti, nei quali lo svolgimento del programma viene fermato, il sistema funziona esattamente come se fosse dotato del suo microprocessore originale.
Una caratteristica fondamentale degli emulatori è che l'emulazione avviene in tempo reale.

Emulatori e simulatori

Non si devono confondere gli emulatori con i simulatori. Mentre i primi operano in ambiente reale, collegati alle altre periferiche del sistema, un simulatore è solamente un programma che gira su di un determinato elaboratore per simulare il comportamento del microprocessore dal punto di vista software, senza tenere conto delle altre periferiche collegate al sistema. Paragonando ad esempio la propria apparecchiatura sotto test ad un aereo, si potrebbe definire l'utilizzo di un emulatore come il pilotaggio del velivolo reale assistito da un istruttore, mentre l'utilizzo di un simulatore si può paragonare al gioco per PC Flight Simulator o ad altri simili. Il vantaggio più importante a favore dei simulatori è evidente: il loro costo è decisamente inferiore. Tra gli svantaggi bisogna segnalare che un simulatore non considera importanti aspetti dell'apparecchiatura che si sta analizzando, quali possono essere gli interrupt, la cache, i refresh della memoria ecc. Riprendendo il paragone proposto, se l'utilizzatore è in grado di pilotare un aereo non avrà certamen-

Menu principale delle opzioni disponibili

```
Assemble  Breakpoint  Display-change  Evaluate   Go  Hw   Init   Load Macro  Nes
Quit      Register    Step      Trace  Unassemble View Window Xfer sYmbol
Replace| Insert
                Atron's 68020 Source Probe, version V3.41
 (C)Copyright Cadre Technologies, Atron Division 1986, 1987, 1988, 1989, 1990
```

Emulatori e simulatori sono strumenti che si complementano con ambiti di impiego diversi

```
Enter breakpoint address:[                                                        ]
═══════════<Enter> to next field: <Tab> to next breakpoint: <Esc> to main menu.

              Breakpoint 0.   Status:<Inactive>

                   ADDRESS OF BREAKPOINT:
<                                                                                 >
To <                                                                              >
Don't care bits:<.... .... .... .... .... .... .... ....>
Memory spaces:   <0,UD,UP,UR,4,SD,SP> {0,UD,UP,UR,4,SD,SP,CPU}

BREAKPOINT VERB:<Execute> {Execute|HWExecute|Read|Write|Fetch|Any}

                  DATA FIELD OF BREAKPOINT:
Data size:       <None> {None|Byte|Word|Long}



                   BREAKPOINT QUALIFIERS:
Logic lines (L3210):<XXXX>                  IPL2,IPL1,IPL0:<XXX>

AFTER TRAP, EXECUTE MACRO/WINDOW KEY:<None>
```

Menu per la definizione di un breakpoint

te problemi nell'utilizzare un simulatore, mentre non è assolutamente certo il contrario: anche se è capace di utilizzare un simulatore non necessariamente è in grado di pilotare un aereo.
La stessa situazione si ripropone tra un simulatore di microprocessore e il sistema reale. Una condizione necessaria, anche se non sufficiente, richiesta ai simulatori perché possano svolgere i compiti previsti, è che i software applicativi funzionino con loro esattamente come in un sistema reale. Con un emulatore invece, si è certamente sicuri che i software applicativi funzionano correttamente come in un sistema reale.

STRUTTURA DI UN EMULATORE

Un emulatore è un dispositivo di dimensioni simili a quelle di una scatola di sardine, collegato ad una pinza di prova che sostituisce il microprocessore emulato. A sua volta questa scatola è collegata all'elaboratore tramite delle schede che devono essere inserite all'interno dello stesso, oppure attraverso la porta seriale. Tutti i modelli disponibili in commercio hanno una struttura che riproduce questa configurazione di base, e che prevede l'utilizzo del calcolatore come interfaccia utente (menu, finestre, ecc.).
Senza dubbio l'elemento più importante di questo sistema è rappresentato dalla pinza di prova.
Questa è formata da un microprocessore identico a quello emulato e da alcuni buffer, cablati su di un circuito molto piccolo e compatto; generalmente viene costruita impilando diversi circuiti stampati realizzati con componenti a montaggio superficiale.
Tramite questi buffer è possibile monitorare l'attività del microprocessore ed eventualmente variarla, simulando ad esempio una lettura dall'apparecchiatura sotto esame mentre in realtà i buffer hanno reindirizzato la stessa in modo che i dati richiesti vengano forniti dall'emulatore.
Per collegare l'emulatore al sistema desiderato è necessario asportare dal sistema stesso il microprocessore e inserire al suo posto la pinza di prova, le cui ridotte dimensioni semplificano l'operazione.
Molto spesso queste pinze di prova sono dotate di zoccoli, per prevenire la rottura dei numerosi terminali di cui sono dotate ed evitare che questi possano toccare i componenti che si trovano vicino alla zona di inserimento del microprocessore nell'apparecchiatura sotto test.
Alcuni emulatori sono forniti dai costruttori assieme ad un piccolo circuito di prova, molto utile e di facile impiego. Questo circuito è dotato di memoria RAM, e consente l'esecuzione di un programma di autodiagnosi per verificare la corretta funzionalità dell'emulatore.

L'elemento più importante e delicato di un emulatore è la pinza di prova

```
Array0: 128K        Array1: 128K        Array2: 128K        Array3: 128K
Don't care bits:<.... .... .... .... XXXX XXXX XXXX XXXX>
Enter new start address of block:[
==========<Tab> to set don't cares: Arrows to move █: <Esc> to main men
 Start      End      Bus    Guarded  Map to   Map Write   Map wait for   Map wai
Address    Address   Size   Access   Array#   Protected   target ready   states
00000000   0001FFFF  Long    No       0         No           No           2
00020000   0003FFFF  Long    Yes      1         Yes          No           2
00040000   FFFFFFFF  Long    No       None
```

Menu di assegnazione dei banchi di memoria interni

UTILIZZO DELL'EMULATORE

Per comprendere ciò che realmente è possibile ottenere da un emulatore è necessario riferirsi ad un esempio concreto. A tal fine si è scelto l'emulatore ATRON, modello PROBE, per emulare il microprocessore 68020 della Motorola.
La sessione inizia digitando:

SOURCE

Ciò fa apparire sullo schermo il menu principale con le seguenti opzioni:

Assemble

Questo emulatore è dotato di un assemblatore di linea che serve per effettuare piccole modifiche sul programma, senza però la pretesa di sostituire un assemblatore esterno.
Selezionando questa opzione sullo schermo ne compaiono altre due: *inserire* o *sostituire*. La prima di queste viene utilizzata per modificare il programma, memorizzando la variazione in una zona di memoria dedicata.
L'emulatore in questo caso inserisce automaticamente un JMP alla nuova zona di memoria che trasla la sequenza di svolgimento del programma alla prima istruzione di questa subroutine.
Con l'opzione sostituire, il codice dell'istruzione esistente viene rimpiazzato dal risultato ottenuto dall'assemblaggio delle nuove istruzioni.

Ciò che rende un emulatore realmente potente sono i breakpoint. Più questi sono avanzati e complessi più l'efficienza aumenta

Breakpoint

Questo comando consente di definire, attivare o eliminare fino a 10 breakpoint. Per definire un breakpoint bisogna assegnargli un numero compreso tra 0 e 9; successivamente appare a video la schermata riportata nella relativa figura. In questa viene richiesto l'inserimento dell'indirizzo di break; questo può essere indicato singolarmente oppure con due valori (To <); in questo secondo caso viene definito un intervallo di indirizzi, compresi i bit di indirizzamento che vengono ignorati. Questa situazione risulta interessante quando l'apparecchiatura sotto prova non è in grado di decodificare tutta la gamma degli indirizzi. L'opzione successiva del menu di definizione consente di selezionare lo spazio di memoria nel quale il breakpoint deve essere eseguito (modalità utente

Pinza di prova con zoccoli di protezione

```
Processor:<Main>

Enter new value:[                                                              ]
═══════════════<Tab> to next field above; Arrows to move █; <Esc> to main menu.
D0=00000000 D4=00000000   A0=00000000 A4=00000000   PC=00000844 CACR=00000000
D1=00000000 D5=00000000   A1=00000000 A5=00000000  USP=00000000 CAAR=00000000
D2=00000000 D6=00000000   A2=00000000 A6=00000000  ISP=00088000  VBR=00000000
D3=00000000 D7=00000000   A3=00000000 A7=00088000  MSP=00000000  SFC=0   DFC=0
SR=2700=T0 S1 M0 I7 X0 N0 Z0 V0 C0
```

Il comando REGISTER consente di visualizzare il contenuto dei registri del microprocessore

o con supervisione del programma o dei dati), facilitando in questo modo l'impostazione di qualsiasi configurazione.
La successiva linea del menu (BREAKPOINT VERB) consente di indicare se il breakpoint deve essere eseguito in lettura, in scrittura o in entrambe le situazioni. Di seguito si trovano le opzioni relative alla conferma dei dati (DATA FIELD OF BREAKPOINT), che permettono di definire il contenuto di un registro o di una posizione di memoria; questo contenuto deve essere soddisfatto per consentire l'esecuzione del break. Inoltre, è possibile indicare la dimensione della parola (byte, word o longword), e ignorare determinati bit se si desidera. Successivamente compare la linea chiamata BREAKPOINT QUALIFIERS. L'emulatore è dotato di quattro *puntali logici*, simili a quelli utilizzati con gli oscilloscopi, che permettono di condizionare l'esecuzione del breakpoint in funzione dello stato logico presente sui puntali stessi.
L'ultima linea è costituita dall'indicazione AFTER TRAP ..., che consente di eseguire una macro o di attivare una finestra nel caso si verifichi il breakpoint.
Anche se è stata fornita una descrizione sufficientemente completa delle funzioni di questo emulatore, vale comunque la pena dimostrare l'enorme potenzialità dei suoi breakpoint, elemento chiave nella fase di analisi degli errori.

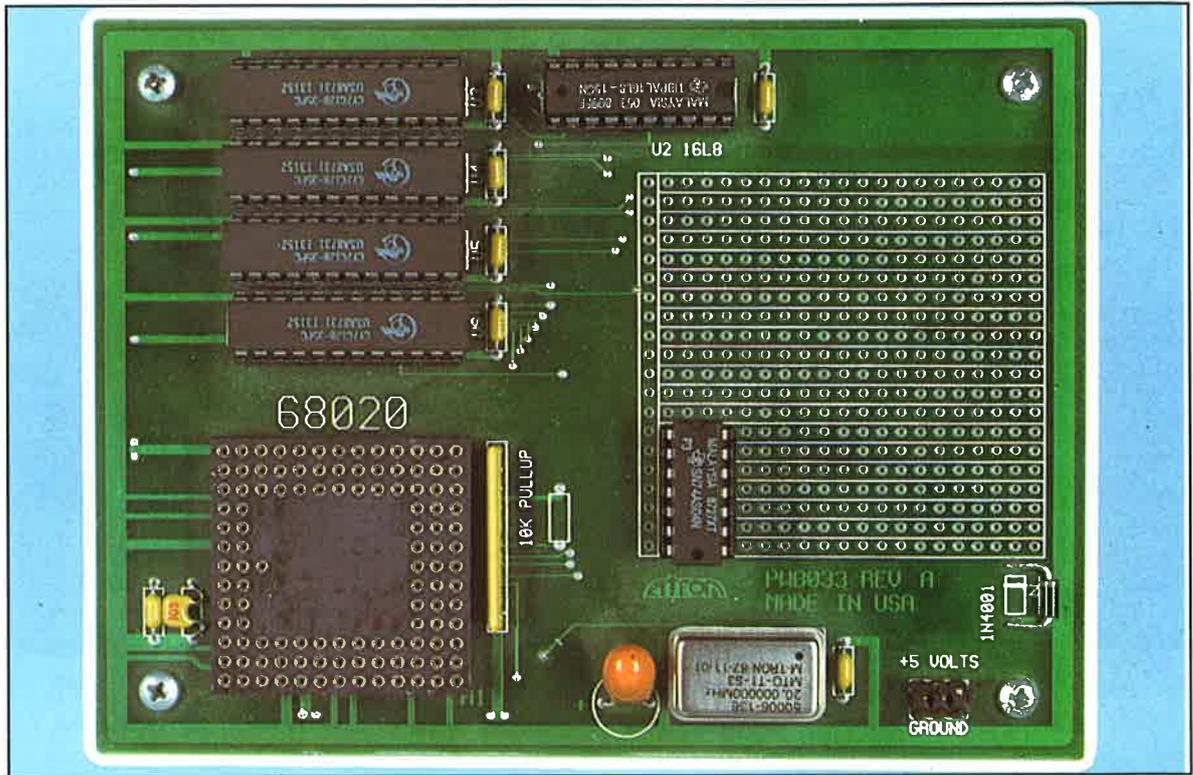
Display-Change
Consente di visualizzare o modificare qualsiasi dato presente in memoria. Per eseguire questo tipo di operazione si può scegliere il formato byte, word, longword oppure virgola mobile. Un aspetto molto interessante di questo emulatore è che è dotato internamente di 512 Kbyte di memoria RAM, ripartita in 4 banchi da 128 Kbyte ciascuno, che possono essere assegnati a qualsiasi intervallo di indirizzi desiderato.
Questi banchi possono sostituire la memoria dell'apparecchiatura sotto test senza che il microprocessore avverta alcuna differenza. Per assegnare questi banchi bisogna indicare, oltre ai parametri convenzionali, anche il numero degli stati di attesa (wait), la protezione dalla scrittura e la modalità di accesso.

Evaluate
Consente di eseguire un'espressione, visualizzandone il risultato in diversi formati.

Go
Inizia l'esecuzione del programma all'indirizzo presente nel contatore del programma.

Hw
Questa opzione consente il controllo del microprocessore per quelle funzioni direttamente relazionate all'hardware, quali:
clock-frequency: misura la frequenza di clock del microprocessore,

Un emulatore molto interessante è quello dotato di banchi di memoria che sostituiscono quelli dell'apparecchiatura sotto prova

Un oscilloscopio può essere utile per la misura dei tempi, solamente se è possibile eseguire cicli di scrittura e lettura infiniti. Per questo motivo questa opzione risulta particolarmente interessante

Circuito di prova per autodiagnosi

execution time: calcola il tempo trascorso tra un comando di "Go" e il successivo breakpoint,
source-breaks-clear: cancella i breakpoint presenti,
interrupt enable: consente o inibisce la comparsa degli interrupt durante l'esecuzione,
DMA-enable: consente o no operazioni di DMA,
halt-line-enable: consente o inibisce l'attivazione del segnale HALT del microprocessore,
break-enable: accetta o ignora i break dovuti a breakpoint,
watchdog-timeout: temporizzazione di errore della durata di 10 ms introdotta automaticamente dall'emulatore,
loop-write-read: esegue un ciclo infinito di lettura e/o scrittura ad un indirizzo prefissato.

Init

Questa opzione permette il salvataggio o il recupero dell'ambiente di lavoro utilizzato la volta precedente, per consentire il proseguimento della prova con le condizioni lasciate.

Load

Facilita il sistema di caricamento di un programma, ed è fornito di una sua tabella dei simboli e di codice sorgente. Accetta molti formati di codici oggetto, quali: S-record, Tektronix, Hex, Binario o COFF-records di UNIX.

Il comando STEP consente di eseguire una o più istruzioni per volta

```
Steps to take for each <Enter>: <0001>          Read operands from memory:<Yes>
After <Enter>, step while: <False>                                   Module: ?
"B" to run to █; "J" to run to instr after █; <Enter> to step from 00000850
══════<Tab> to next field above; PgUp/Dn, Arrows to move █; <Esc> to main menu.
D0=00000000 D4=00000000   A0=00000000 A4=00000000   PC=00000850 CACR=00000000
D1=00000000 D5=00000000   A1=00000000 A5=00000000  USP=00000000 CAAR=00000000
D2=00000000 D6=00000000   A2=0009A060 A6=00000000  ISP=00088000  VBR=00000000
D3=00000000 D7=00000000   A3=00000000 A7=00088000  MSP=00000000  SFC=0   DFC=0
SR=2704=T0 S1 M0 I7 X0 N0 Z1 V0 C0

00000844 CLR.l       D0
00000846 MOVEC.l     D0,CACR
0000084A MOVE.l      #0009A060,A2
00000850 MOVE.w      #018F,(00000002,A2)
 ^ ^ ^ ^       Op 2 address=0009A062
00000856 MOVE.w      #6000,(A2)
0000085A MOVE.w      #FFFF,(00000008,A2)
00000860 MOVE.w      #FFFF,(0000000A,A2)
00000866 MOVE.w      #0000,A6
0000086A MOVE.w      #2000,SR
0000086E CLR.l       (0008002A)
00000874 CLR.l       (0008001C)
0000087A CLR.l       (00080022)
00000880 CLR.w       (00080020)
00000886 CLR.w       (00080026)
```

Macro

Sono file che contengono una sequenza di istruzioni che vengono memorizzate e possono essere eseguite ripetutamente premendo un solo tasto. É possibile definire anche delle macro condizionali, con istruzioni tipo If, Loop e While.

Nest

Questo comando consente di analizzare il contenuto dello "stack" del

```
Search address:<Any>                    Space:<Any>   Verb:<Any>     Data:<Any>
<.... .... .... .... .... .... .... ....>
Enter memory space to search for:[Any] {Any|0  |UD |UP |UR |4  |SD |SP |CPU}
═══<Space> for next choice: <Enter> or <Tab> to next field:  <Esc> to main menu
<Home> to trace counter at cycle #0032
Cycle Address Spc  Data     Strb RW If Rn Ep Bk  BP  Rs P G BgBe Log G24  G32
 0000 00087FF1 5  90FFFFFF  1111 1  0  1  0  0  0000 1  0 0 0 1   F  4B   BE
 0001 00000B90 6  0A50FFFF  0000 1  1  1  0  0  0000 1  0 0 0 1   F  08   B6
 0002 00000B94 6  484FB443  0000 1  1  1  0  0  0000 1  0 0 0 1   F  08   B6
 0003 FFFFFFFB 7  FFFFFEFF  1111 1  0  1  0  0  0000 1  0 0 0 1   F  5F   AE
 0004 00087FEC 5  20082008  0011 0  0  1  0  0  0000 1  0 0 0 1   F  9D   BE
 0005 00000074 5  0000064E  0000 1  0  1  0  0  0000 1  0 0 0 1   F  1C   BE
 0006 00087FEE 5  00000000  0011 0  0  1  0  0  0000 1  0 0 0 1   F  1D   BE
 0007 00087FF0 5  0B900B90  0011 0  0  1  0  0  0000 1  0 0 0 1   F  9D   BE
 0008 0000064C 6  4E7333F9  0000 1  1  1  0  0  0000 1  0 0 0 1   F  1C   BE
 0009 00000650 6  00080036  0000 1  1  1  0  0  0000 1  0 0 0 1   F  1C   BE
 000A 00087FF2 5  00740074  0011 0  0  1  0  0  0000 1  0 0 0 1   F  9D   BE
 000B 00080036 5  80000074  0011 1  0  1  0  0  0000 1  0 0 0 1   F  9D   BE
 000C 00000654 6  00080034  0000 1  1  1  0  0  0000 1  0 0 0 1   F  1C   BE
 000D 00000658 6  33C00008  0000 1  1  1  0  0  0000 1  0 0 0 1   F  1C   BE
 000E 00080034 5  80008000  0011 0  0  1  0  0  0000 1  0 0 0 1   F  9D   BE
 000F 0000065C 6  00364E73  0000 1  1  1  0  0  0000 1  0 0 0 1   F  1C   BE
 0010 00080036 5  80008000  0011 0  0  1  0  0  0000 1  0 0 0 1   F  9D   BE
 0011 00000660 6  53390008  0000 1  1  1  0  0  0000 1  0 0 0 1   F  1C   BE
 0012 00087FFC 5  20080008  0011 1  0  1  0  0  0000 1  0 0 0 1   F  9D   BE
```

L'opzione TRACE consente di visualizzare in dettaglio l'attività del microprocessore in ciascun ciclo

microprocessore, per determinare dove e come sono state eseguite chiamate a subroutine, quali sono i parametri trasferiti a queste e quali sono gli indirizzi di ritorno.

Quit

Tramite questo comando si chiude la sessione di emulazione.

Register

Consente di modificare tutti i registri del micro emulato, compresi il contatore di programma, lo stack pointer, la tabella degli interrupt, ecc.

Step

Con questo comando l'esecuzione delle istruzioni viene effettuata una ad una o a gruppi. Consente inoltre di selezionare se queste sono in codice sorgente o in codice oggetto.

Trace

Visualizza il codice dell'istruzione che è stata eseguita prima dell'ultimo breakpoint. Consente inoltre di scegliere tra una visualizzazione dell'attività ciclo per ciclo o istruzione per istruzione.

Unassemble

Esegue un disassemblaggio della porzione di memoria indicata; questo può essere simbolico, per cui vengono visualizzate anche le etichette e le istruzioni del codice sorgente quando si opera con questo formato.

View

Visualizza sullo schermo il file del codice sorgente o qualsiasi altro file di testo. Inoltre, è possibile mantenere aperti fino a 10 file, strutturati in modo che dopo lo spostamento da uno all'altro si ritorna nella posizione abbandonata con l'ultimo spostamento; ciò consente di muoversi con grande agilità da un file all'altro.

Window

Consente di creare una finestra sullo schermo che contiene una informazione qualsiasi. Per poter eseguire questa funzione si deve definire preventivamente la sequenza di tasti che attivano

I breakpoint e l'esecuzione passo passo rappresentano il 90% delle potenzialità di un emulatore

Il comando UNASM disassembla il contenuto della memoria

```
Start address:  <00000844          Memory space:<SP >
Display instruction words:<Yes>   Display operand addresses and values:<Yes>
Enter number of instructions:[                                              ]
═══PgUp/PgDn/Arrows move within memory: <Tab> to next field: <Esc> to main menu
00000844 4280
 ^ ^ ^ ^ CLR.l      D0
00000846 4E7B0002
 ^ ^ ^ ^ MOVEC.l    D0,CACR
 ^ ^ ^ ^      Op 1 value=00000000
0000084A 247C0009A060
 ^ ^ ^ ^ MOVE.l     #0009A060,A2
00000850 357C018F0002
 ^ ^ ^ ^ MOVE.w     #018F,(00000002,A2)
 ^ ^ ^ ^      Op 2 address=00000002
00000856 34BC6000
 ^ ^ ^ ^ MOVE.w     #6000,(A2)
 ^ ^ ^ ^      Op 2 address=00000000
0000085A 357CFFFF0008
 ^ ^ ^ ^ MOVE.w     #FFFF,(00000008,A2)
 ^ ^ ^ ^      Op 2 address=00000008
00000860 357CFFFF000A
 ^ ^ ^ ^ MOVE.w     #FFFF,(0000000A,A2)
 ^ ^ ^ ^      Op 2 address=0000000A
00000866 3C7C0000
 ^ ^ ^ ^ MOVE.w     #0000,A6
```

```
Steps to take for each <Enter>: <0001>            Read operands from memory:<Yes>
After <Enter>, step while: <False>                                      Module: ?
"B" to run to █; "J" to run to instr after █; <Enter> to step from 00000844
═════<Tab> to next field above: PgUp/Dn, Arrows to move █; <Esc> to main menu.
D0=00000000 D4=00000000   A0=00000000 A4=00000000   PC=00000844 CACR=00000000
D1=00000000 D5=00000000   A1=00000000 A5=00000000  USP=00000000 CAAR=00000000
D2=00000000 D6=00000000   A2=00000000 A6=00000000  ISP=00088000  VBR=00000000
D3=00000000 D7=00000000   A3=00000000 A7=00088000  MSP=00000000  SFC=0  DFC=0
SR=2700=T0 S1 M0 I7 X0 N0 Z0 V0 C0

<AltZ>
 0008 8000 0000 0844 0000

00000844 CLR.l       D0
00000846 MOVEC.l     D0,CACR
0000084A MOVE.l      #0009A060,A2
00000850 MOVE.w      #018F,(00000002,A2)
00000856 MOVE.w      #6000,(A2)
0000085A MOVE.w      #FFFF,(00000008,A2)
00000860 MOVE.w      #FFFF,(0000000A,A2)
00000866 MOVE.w      #0000,A6
0000086A MOVE.w      #2000,SR
0000086E CLR.l       (0008002A)
00000874 CLR.l       (0008001C)
0000087A CLR.l       (00080022)
```

L'opzione WINDOW consente di aprire delle finestre per visualizzare il contenuto delle variabili

la finestra, la sua posizione sullo schermo e il suo contenuto, che può essere costituito da una espressione, dai dati presenti in una zona di memoria, da una catena di caratteri oppure da una etichetta. Questa opzione è molto utile quando si esegue un programma istruzione per istruzione, per visualizzare simultaneamente sullo schermo il contenuto di una determinata zona di memoria. Purtroppo il contenuto della finestra non viene rinfrescato dinamicamente, ma viene elaborato ogni volta che viene richiamato. Per questa ragione la funzione delle finestre (è consentita la presenza di più finestre sullo schermo) è solo quella di ricordare il contenuto di una variabile in un istante precedente.

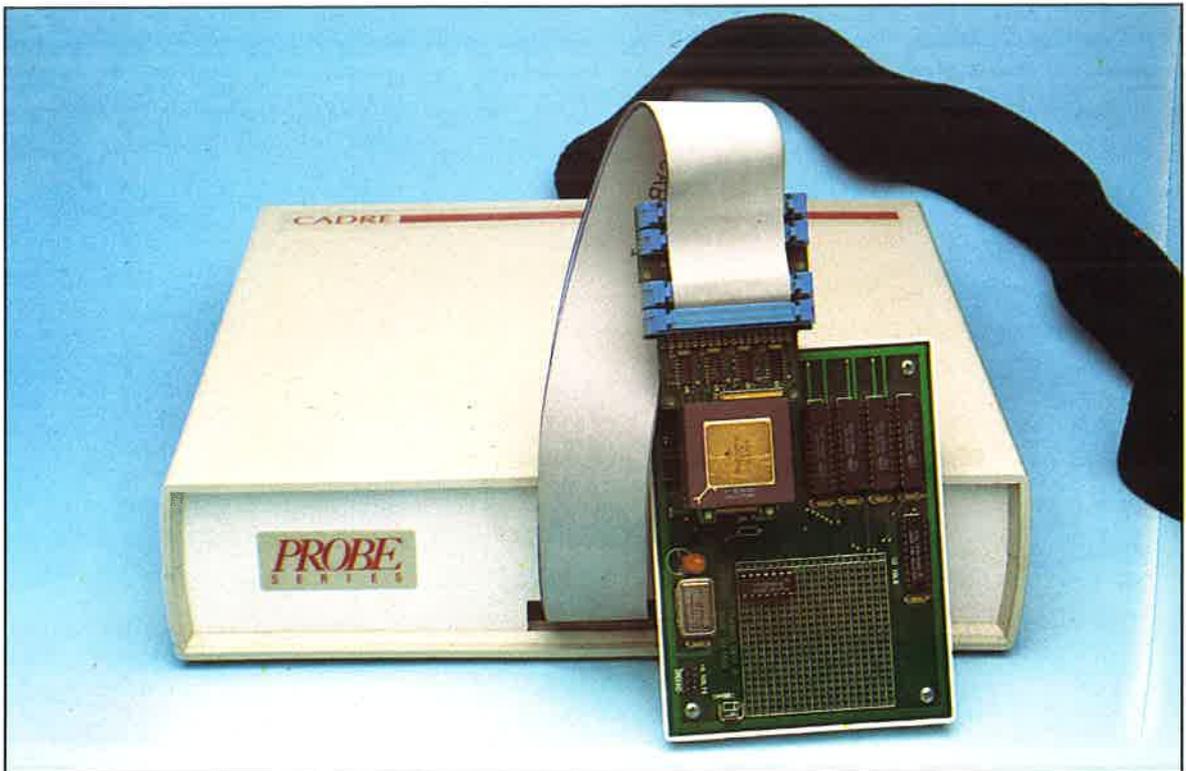
Xfer

Questa opzione consente di eseguire diverse operazioni su un blocco di memoria specificato, quali memorizzare questo blocco in un file, confrontarlo, muoverlo oppure inizializzarlo ad un valore predeterminato.

Symbol

La tabella dei simboli viene generata contemporaneamente al codice oggetto durante la compilazione, essendo necessaria per la depurazione simbolica di un programma. Questa depurazione viene eseguita nel linguaggio ad alto livello con cui il programma è stato scritto, e con i nomi delle variabili e delle etichette che sono state utilizzate in quella fase. L'opzione Symbol consente di realizzare le operazioni associate alla tabella dei simboli, quali il caricamento, la modifica e la cancellazione.

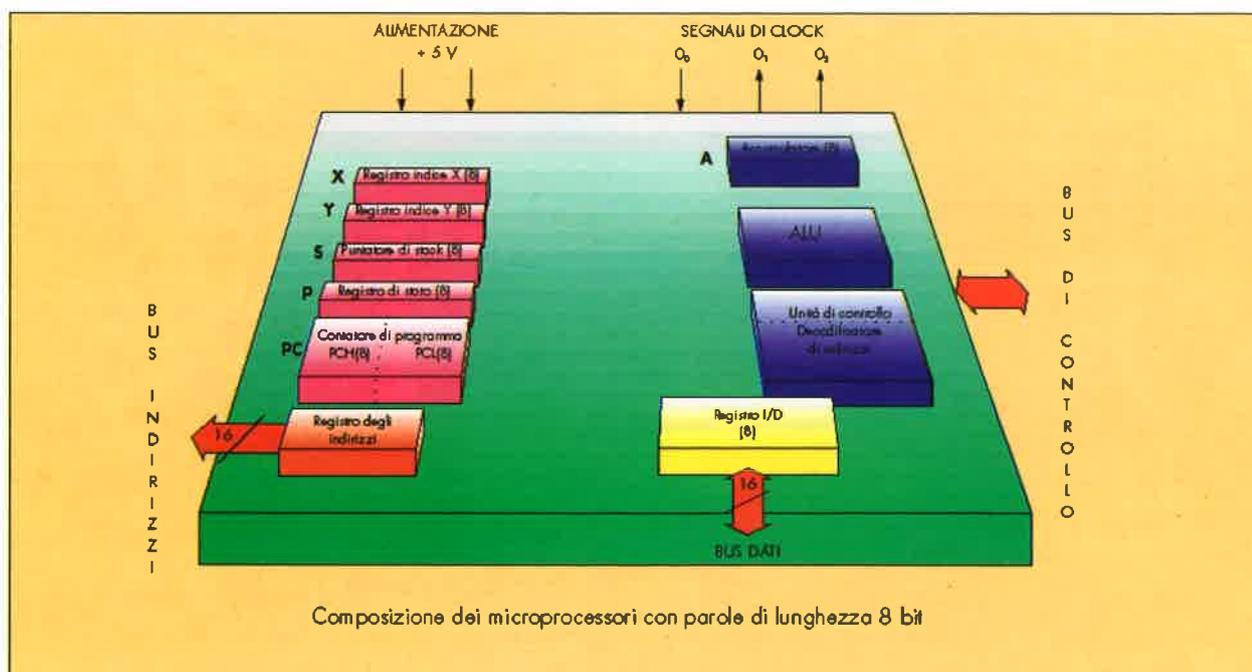
Queste sono tutte le opzioni disponibili con questo emulatore. Qualsiasi altro modello esegue fondamentalmente le stesse funzioni, tenendo presente però che esistono alcune ovvie differenze tra i diversi costruttori.

Collegamento dell'emulatore con il circuito di prova

Un assemblatore e un disassemblatore di linea sono indispensabili in qualsiasi emulatore

ARCHITETTURA DI UN MICROPROCESSORE

Dopo aver descritto le diverse applicazioni dei microprocessori, e in modo particolare quelle specifiche per gli elaboratori, verranno esaminate nei prossimi capitoli le strutture interne, i collegamenti e i circuiti ausiliari di alcune loro famiglie.

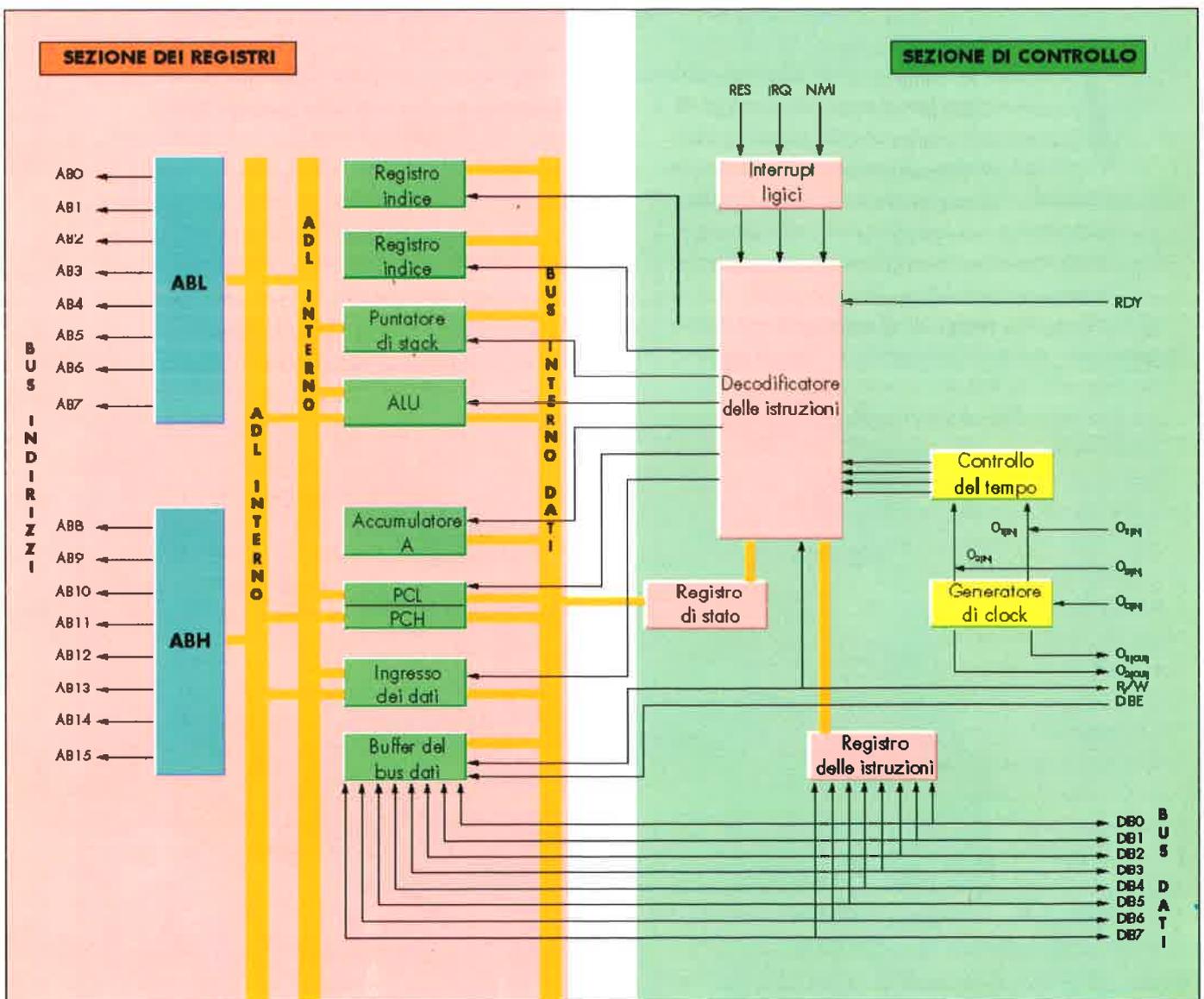
I microprocessori disponibili in commercio si possono suddividere in diverse famiglie, che a loro volta possono essere classificate secondo un parametro di riferimento costituito dalla lunghezza della parola che sono in grado di elaborare. Fondamentalmente le famiglie sono tre: quella dei microprocessori a 8 bit, a 16 bit e a 32 bit.
Si inizia questo viaggio nelle famiglie dei microprocessori partendo da quella a 8 bit, in quanto è la più elementare e semplice da comprendere. Le altre famiglie verranno analizzate nei capitoli successivi.
Anche se ogni fabbricante costruisce i propri microprocessori con una determinata architettura, la struttura fondamentale di tutti quelli presenti in commercio è simile al punto che, conoscendone uno per ciascuna famiglia, è possibile utilizzare gli altri della stessa famiglia con estrema semplicità.

Composizione dei microprocessori con parole di lunghezza 8 bit

INTRODUZIONE AI MICROPROCESSORI A 8 BIT

In ciascuna famiglia esistono parecchi circuiti differenti, utilizzati soprattutto negli elaboratori, che vengono costruiti generalmente in tecnologia NMOS. Ad esempio, tra i microprocessori a 8 bit se ne possono trovare alcuni controllati con segnali di clock esterni e con una capacità di memoria che può arrivare a 64 Kbyte. In funzione delle diverse versioni ogni contenitore è dotato di un numero diverso di terminali di collegamento; di questi si parlerà successivamente. Le caratteristiche fondamentali di questi microprocessori possono essere riassunte in dieci punti:

1. *alimentazione + 5 volt,*
2. *capacità di memoria fino a 64 Kbyte,*
3. *aritmetica binaria e decimale,*
4. *diversi tipi di interrupt,*
5. *fino a 56 istruzioni diverse,*
6. *13 modalità di indirizzamento,*
7. *segnale di clock interno ed esterno,*
8. *contenitori da 40 e 28 terminali,*
9. *frequenze di lavoro di 1 o 2 MHz,*
10. *elaborazione di parole da 8 bit.*

Per quanto riguarda la velocità di elaborazione del microprocessore, si deve tener presente che a frequenze più alte corrisponde un maggiore riscaldamento, e questo a volte può tradursi in guasti o errori nel sistema che si sta progettando.

Schema interno di un microprocessore, nel quale si possono osservare i diversi collegamenti tra i diversi elementi

Struttura dei bit di un registro di stato

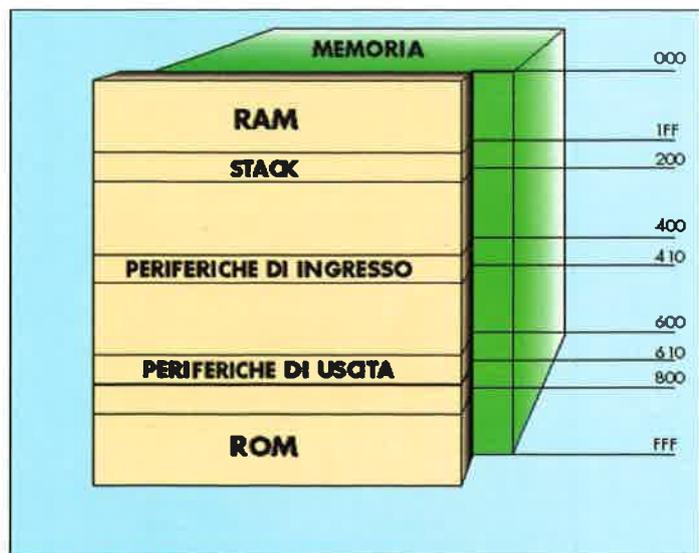
Nei microprocessori con frequenza 1 MHz la durata di quello che viene definito ciclo macchina è di 1 microsecondo; poiché l'istruzione più rapida richiede almeno due cicli macchina, il suo tempo di esecuzione è di 2 microsecondi. Per una velocità di 2 MHz il ciclo macchina è di 500 nanosecondi, e di conseguenza l'esecuzione della stessa istruzione richiede 1 microsecondo. Allo stesso modo, l'istruzione più lunga richiede 7 cicli macchina, per cui nel primo caso la sua esecuzione avrà una durata di 7 microsecondi, mentre nel secondo di 3,5 microsecondi. Con i tempi di risposta indicati, i microprocessori di questa famiglia possono operare con delle velocità accettabili, che vengono ulteriormente ottimizzate grazie ad una struttura chiamata "pipe-line" che consente di eseguire diverse operazioni contemporaneamente. Questo tipo di architettura è caratterizzata dal fatto che è possibile iniziare nuove operazioni senza tenere conto dell'eventuale presenza di altri compiti precedenti e senza il bisogno di concluderli; vale a dire che consente di sovrapporre delle istruzioni senza dover attendere che qualcuna di queste debba essere eseguita.

STRUTTURA DEI MICRO A 8 BIT

L'architettura interna di questo tipo di microprocessori è molto simile a quella generale descritta nei capitoli precedenti.
Il microprocessore è costituito da:
- *contatore di programma,*
- *registro degli indirizzi,*
- *registro delle istruzioni,*
- *registro dei dati,*
- *unità di controllo,*
- *decodificatore delle istruzioni,*
- *unità aritmetico-logica,*
- *accumulatore,*
- *registri ausiliari,*
- *bus di indirizzamento,*
- *bus di controllo.*

Per quanto concerne il bus di indirizzamento, è necessario segnalare che esiste un solo bus comune per la memoria che contiene i dati e per quella che contiene le istruzioni. Anche se il lettore è già a conoscenza delle funzioni che vengono eseguite dagli elementi di un microprocessore, è comunque opportuno analizzare gli aspetti che differenziano questa famiglia dalle altre.
Il decodificatore delle istruzioni è in grado di ricevere ed interpretare parole di 8 bit di lunghezza, per cui può accettare fino a 256 istruzioni diverse. Poiché ognuna di queste può sfruttare codici differenti, nel caso in esame ne sono necessarie solamente 56 per un totale di 160 codici di istruzione.
Il bus di indirizzamento della memoria consente la selezione di tutte le celle di memoria, e il numero delle linee che lo compongono determina la capacità della stessa. Di conseguenza, nel caso di un bus a 16 linee, si possono controllare 64 Kbyte di celle da 8 bit ciascuna, per cui è possibile indirizzare fino a 65.536 indirizzi.
Il bus di controllo trasporta i segnali di sincronizzazione e di controllo ai circuiti ausiliari che appartengono al sistema.
I registri ausiliari possono essere di tre diversi tipi:
- *registri indice,*
- *registro di stato,*
- *stack pointer.*

I registri indice si suddividono a loro volta in *registri indice X* e *registri indice Y*. Entrambi servono per realizzare una modalità di indirizzamento definita "indicizzata", con la quale l'individuazione dell'operando di una istruzio-

Paginazione della memoria di un microprocessore

ne viene ricercato nella memoria aggiungendo il contenuto di questi registri all'indirizzo specificato nell'istruzione.

Il registro di stato, o registro P, è un registro a 8 bit, definiti anche *flag*, sette dei quali sono determinanti e servono per memorizzare la condizione del sistema durante l'esecuzione di una determinata istruzione.

Il puntatore di stack indirizza la pagina 1 della memoria come se questa avesse una struttura a pila LIFO

Di seguito questi bit, la cui struttura è rappresentata nella figura corrispondente, vengono esaminati singolarmente, tenendo presente che ciascuno di loro può assumere due significati diversi in funzione dello stato logico nel quale si trova in un determinato istante.

Flag di segno (N): questo flag indica il segno del risultato dell'operazione eseguita, 1 se è negativo e 0 se è positivo.

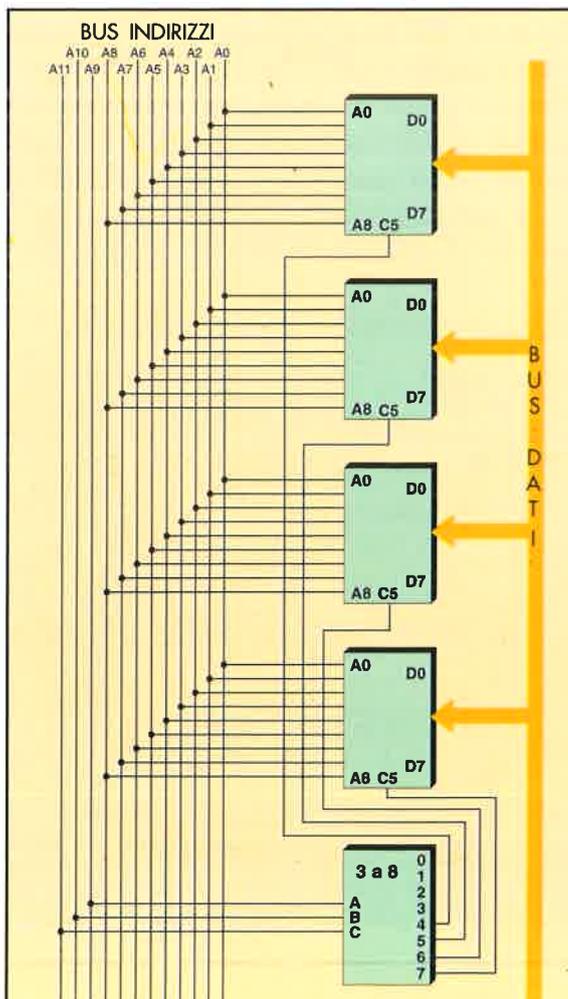
Flag di overflow (V): determina se esiste o meno il riporto sul settimo bit quando si esegue una operazione con due parole da 8 bit.

Flag di riporto (C): se non si utilizza il bit di segno, C passa a 1 quando vi è un riporto nell'ottavo bit, vale a dire quando il risultato dell'operazione supera il valore esadecimale FF.

Flag zero (Z): commuta a zero quando il risultato dell'operazione precedente è zero.

Flag per il trattamento aritmetico (D): quando è a livello logico 1 l'unità aritmetico-logica esegue operazioni in decimale o BCD, mentre se è a livello logico zero esegue operazioni in binario o esadecimale.

Flag degli interrupt (I): i microprocessori di questi famiglia hanno due tipi di interrupt, conosciuti con i nomi di *mascherabile* e *non mascherabile*. Quando il flag è a 1 non vengono accettate le istruzioni mascherabili, poiché queste vengono riconosciute solamente quando I è a 0.

Flag break (B): questo flag assume valore 1 quando l'interrupt mascherabile ammesso dal microprocessore è stato generato dal software con l'istruzione BRK. Se l'interrupt è stato generato dall'hardware il flag commuta a 0. L'ultimo registro viene chiamato "puntatore di stack" o *Stack Pointer*. Per capire il suo compito si deve considerare la memoria gestita come se fosse paginata; ogni posizione viene suddivisa in gruppi di 256 posizioni, ciascuno dei quali forma quelle comunemente conosciute con il nome di pagine. Poiché le pagine 0 e 1 di una memoria hanno un carattere molto particolare, nel caso della pagina 1 l'indirizzamento avviene attraverso il puntatore di stack. Questo è costituito da un contatore a 8 bit che indirizza la pagina 1 della memoria come se si trattasse di una struttura a pila tipo LIFO.

Convertitore bidirezionale serie-parallelo

La maggior parte dei circuiti presentati finora vengono controllati attraverso la porta parallela centronics del calcolatore. Quello che viene proposto di seguito non richiede invece alcun controllo, poiché si tratta di un convertitore bidirezionale seriale-parallelo.

La maggior parte delle realizzazioni controllate esternamente prevedono lo scambio delle informazioni in formato parallelo

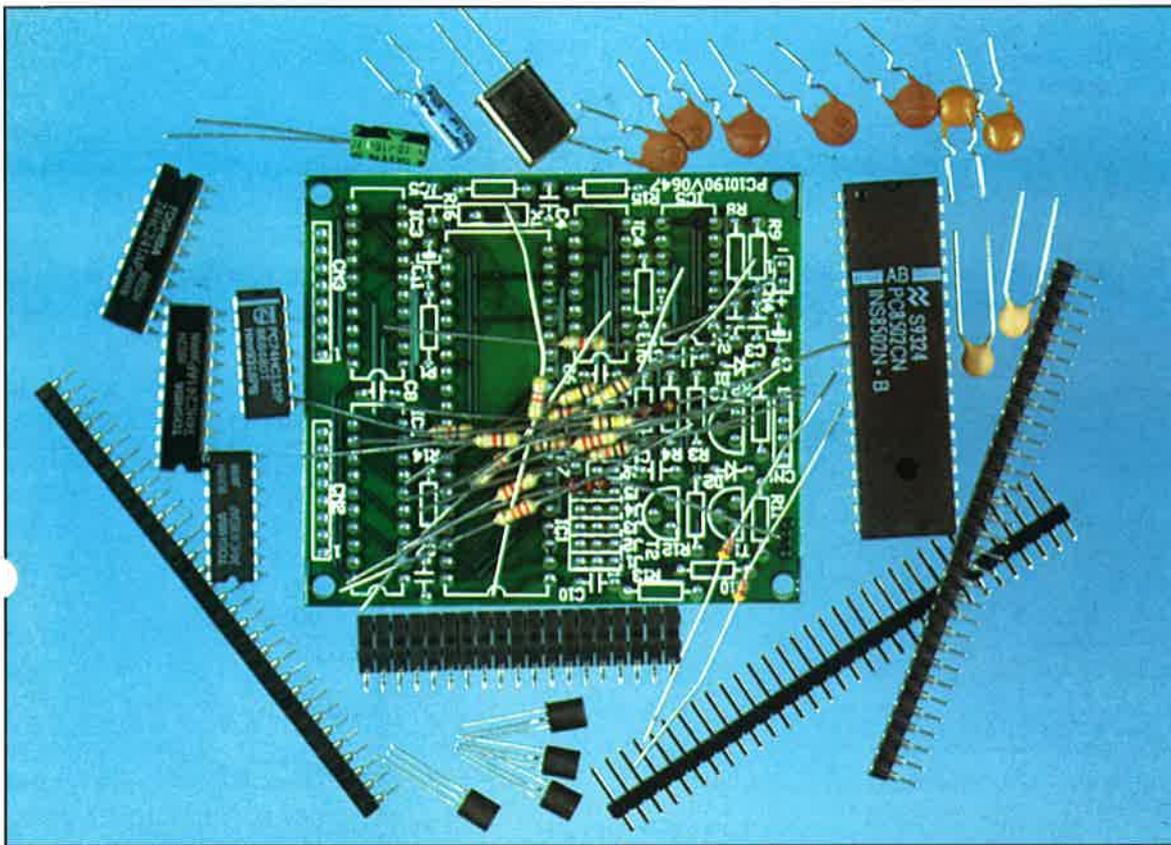
Normalmente lo scambio di informazioni tra il calcolatore e il mondo esterno è limitato a una porta seriale e a una porta parallela, per cui è possibile controllare un solo dispositivo esterno di ciascun tipo. Generalmente la maggior parte dei dispositivi controllati esternamente scambiano le informazioni con l'elaboratore in modalità parallela, poiché la gestione di questo protocollo è più

Come si può verificare nello schema elettrico, il cuore del convertitore bidirezionale è un UART

L'UART ha la funzione di convertire i dati seriali che arrivano attraverso il suo ingresso RSI in un dato parallelo che presenta sulle sue uscite da TD0 a TD7

semplice in quanto non richiede la conversione dei dati da parallelo a seriale e viceversa. I problemi iniziano quando è necessario controllare due dispositivi dello stesso tipo contemporaneamente.
Si supponga, ad esempio, di voler progettare un dispositivo controllato esternamente nel quale lo scambio dei dati con il calcolatore avvenga attraverso l'interfaccia parallela, poiché questa risulta più semplice da realizzare e richiede un minor numero di componenti. Se alla porta centronics del calcolatore è già collegata una stampante, e si desiderano stampare i risultati del processo di controllo, non è possibile collegare il proprio dispositivo a questa porta.
Il convertitore bidirezionale che viene proposto risolve questo problema, poiché permette di progettare il proprio dispositivo controllato esternamente con una modalità di interscambio dei dati parallela, senza la necessità di dover installare una ulteriore interfaccia LPT2 per comunicare con il calcolatore. Il convertitore agisce infatti come una interfaccia tra la porta seriale del calcolatore e il dispositivo esterno, trasformando i dati seriali provenienti dal calcolatore in dati paralleli che possono essere utilizzati dal dispositivo, e viceversa.

IL CIRCUITO

Il convertitore bidirezionale è basato su di un *UART (Universal Asynchronous Receiver/ Transmitter)*. Questo circuito integrato ha il compito di convertire i dati che arrivano al suo ingresso RSI in formato seriale, in un dato in parallelo che presenta sulle uscite TD0-TD7; inoltre, è in grado di eseguire il processo inverso con il dato paralle-

Componenti necessari per realizzare il dispositivo

Lo scorrimento seriale dei bit è controllato dal clock a 19200 Hz

lo che entra sugli ingressi RD0-RD7, presentandolo in formato seriale sull'uscita TSO. Inoltre, permette il controllo del protocollo di comunicazione, della velocità di trasmissione e del numero di bit dei dati, di parità e di stop.
Questo componente viene proposto da diversi fabbricanti, per cui può essere reperito in commercio con sigle diverse; ad esempio AY-5-1013 della General Instruments, CPD1854 della RCA, oppure 8502 della Intel.
Lo scorrimento dei bit seriali è controllato dal clock a 19.200 Hz applicato agli ingressi RCP (clock di ricezione) e TCP (clock di trasmissione). Questa frequenza fissa la velocità di trasmissione (in entrambi i versi di comunicazione) dell'interfaccia a 1.200 baud (19.200/6). Il clock utilizzato è gestito da un divisore binario per 128 (74HC4060) al quale viene collegato un oscillatore, controllato da un cristallo al quarzo da 2,4576 MHz.
L'ingresso (da RD0 a RD7) e l'uscita (da TD0 a TD7) parallela comunicano con il dispositivo che si desidera controllare attraverso i buffer IC2 e IC3 (74HC241); questi isolano l'UART dalla circuiteria esterna, per cui è possibile controllare qualsiasi dispositivo che si desidera collegare.

Inizialmente si devono eseguire i collegamenti tra le due facce dello stampato, e successivamente si possono montare le resistenze e gli zoccoli per gli integrati

Dopo le resistenze si devono montare i condensatori

I dati seriali ricevuti sul terminale RXD del connettore CN1 vengono invertiti, inviati al terminale RSI di IC1 e presentati sull'uscita parallela, mentre i dati presenti sull'ingresso parallelo vengono convertiti in formato seriale e inviati in uscita al terminale TXD di CN1 tramite il terminale TSO, a valle del quale vengono invertiti.
Il formato dei dati viene definito per mezzo dei jumper di programmazione collegati ai terminali 35-39 dell'UART, che consentono di impostare la loro lunghezza, il numero dei bit di stop e la parità. Tramite la tabella riportata nella figura corrispondente è possibile programmare il convertitore nel modo più consono alle proprie esigenze.

Il segnale TEOC, fine della trasmissione dei caratteri (Trasmitter End Of Character), viene utilizzato per generare il segnale di protocollo RTS

Gli altri elementi circuitali hanno il compito di generare e controllare i segnali di protocollo necessari per il funzionamento dell'interfaccia.
Quando la parola completa è stata trasferita nel registro di ricezione, il segnale RDA (*Received Data Available - dato ricevuto disponibile*) commuta a livello alto; di conseguenza il segnale /RDAR (*Received Data Available Reset - reset del dato ricevuto*, attivo a livello basso), collegato al primo tramite l'invertitore T4, commuta a livello basso innescando l'inizializzazione del ricevitore dell'UART. Se il ponticello J6 è inserito, il segnale RDA controlla anche l'ingresso /TDS (*Transmitter Data Strobe - conferma del dato trasmesso*, attivo a livello basso), facendo in modo che venga caricato un nuovo dato parallelo (da TD0 a TD7) nel registro di trasmissione. Questo ponticello consente di utilizzare il segnale di protocollo CTS (*Clear To Send*).
Il segnale TEOC (*Transmitter End Of Character - termine della trasmissione dei caratteri*) viene utilizzato per generare il segnale di protocollo RTS e controllare, unitamente a CTS, l'ingresso /TDS.
L'attivazione del segnale CTS indica all'UART che deve inviare un nuovo dato seriale. Per evitare conflitti tra i segnali CTS e TEOC si utilizza il bistabile R-S, composto da IC5A e IC5B. Quando si applica la tensione di alimentazione (5 V),

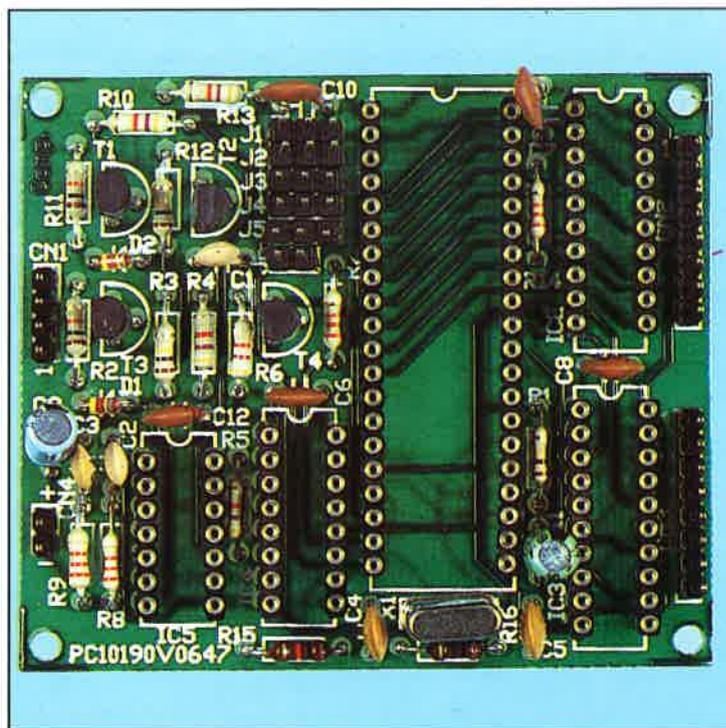
Gli zoccoli devono essere saldati su entrambe le facce

Dopo gli zoccoli si devono saldare le resistenze

l'UART resetta tutti i suoi registri grazie alla rete RC collegata al suo ingresso di attivazione (MR). I segnali TSO e TEOC vengono impostati a livello alto, mentre RDA si porta a livello basso. Se il ponticello J6 non viene inserito, l'impulso del segnale TEOC viene applicato invertito all'ingresso /TDS, e ha inizio il processo di trasmissione.

MONTAGGIO E VERIFICA

Prima di procedere al montaggio del circuito è consigliabile classificare tutti i componenti in modo da evitare qualsiasi errore nei valori degli stessi. Dopo aver selezionato il materiale, e con il piano di assemblaggio bene in vista, si può procedere al montaggio.
Poiché il circuito stampato è a doppia faccia con fori non metallizzati, è necessario saldare su entrambe le facce dello stampato tutte quelle isole alle quali è collegata una pista dal lato componenti, poiché altrimenti non vi sarebbe continuità tra una faccia e l'altra dello stampato e il dispositivo non potrebbe funzionare. Si consiglia di utilizzare per i circuiti integrati degli zoccoli con terminali torniti, per evitare di saldare direttamente il componente sullo stampato e di conseguenza prevenire i possibili problemi legati al loro surriscaldamento. Gli zoccoli sono i primi componenti che devono essere montati sullo stampato, mentre gli integrati devono essere inseriti solo al termine di tutte le operazioni di saldatura. Successivamente si possono montare le resistenze da R1 a R16, i condensatori ceramici C10, C12 e da C1 a C8, e i connettori CN1, CN2, CN3, CN4, J1, J2, J3, J4, J5 e J6. Per i connettori è necessario tagliare due file da due terminali per CN4 e J6, cinque file da tre terminali per i connettori da J1 a J5, una da 4 terminali per CN1 e due file da otto terminali per CN2 e CN3. Si prosegue con i condensatori elettrolitici C9 e C11 e con i diodi D1 e D2, prestando particolare

I connettori possono essere saldati per ultimi

Poiché il circuito stampato è a doppia faccia con fori non metallizzati, è necessario saldare i componenti su entrambe le facce

Dopo aver terminato il montaggio bisogna controllarlo visivamente

attenzione al loro orientamento, che deve corrispondere alla polarità riportata sulla serigrafia dello stampato. I transistor da T1 a T4 sono identici e anche in questo caso, quando vengono montati, bisogna rispettare la posizione e l'orientamento indicati sulla serigrafia. Infine si devono inserire gli integrati nei rispettivi zoccoli, come al solito rispettando il riferimento indicato sulla serigrafia. Dopo aver ultimato tutte le operazioni di montaggio dei componenti è opportuno controllare il circuito visivamente, verificando che tutti i componenti siano montati nella posizione corretta e con la polarità indicata. Inoltre, si devono controllare le saldature, verificando che siano state eseguite tutte, che siano di buona qualità, e che non ci siano dei cortocircuiti tra piste o isole adiacenti.

L'UART è il componente incaricato di convertire i dati dal formato seriale a quello parallelo e viceversa

AVVIAMENTO DEL CIRCUITO

Con il circuito correttamente montato si può procedere alla verifica del suo funzionamento. Si deve quindi applicare la tensione di alimentazione (5 V) al connettore CN4, controllandone successivamente la presenza sui terminali corrispondenti degli integrati.
Senza alimentare il circuito si devono impostare i ponticelli in modo che il formato dei dati sia a 8 bit, senza parità e con due bit di stop, in accordo con la tabella di configurazione riportata nella

I "buffer" isolano dall'esterno l'ingresso e l'uscita parallela, fornendo la corrente necessaria per pilotare la circuiteria esterna

figura corrispondente. Per eseguire il collegamento tra il convertitore e la porta seriale del calcolatore si deve utilizzare un cavo di collegamento seriale RS-232 (consultare il manuale delle caratteristiche tecniche del PC).

La verifica deve essere eseguita in due diversi modi, uno con il protocollo e l'altro senza. Per la prova con il protocollo si deve lasciare il ponticello J6 aperto e, quando si realizza il cavo, si deve eseguire un ponticello tra i segnali DSR e DTR sul connettore tipo D (terminali 6 e 20 se il connettore è un DB25, terminali 6 e 4 se il connettore è un DB9). Per effettuare la verifica senza protocollo il ponticello J6 deve essere invece cortocircuitato, e sul connettore si devono ponticellare, oltre ai segnali DSR e DTR, i segnali RTS e CTS (terminali 4 e 5 rispettivamente per il connettore DB25, e terminali 7 e 8 per il connettore DB9).

Esempio di circuiteria che può essere utilizzata per controllare un qualsiasi dispositivo

Poiché inizialmente non c'è un dispositivo da controllare con questo convertitore, per eseguire una verifica funzionale si collegano i bit dell'uscita all'ingresso parallelo, in modo che il dato inviato dal calcolatore risulti predisposto per essere spedito dal convertitore. Per questa verifica è necessario un software che invii e riceva i dati attraverso la porta seriale, e che li visualizzi sullo schermo. Un programma che può essere utilizzato per questo scopo, scritto in PASCAL, può essere il seguente:

Il programma invia ciclicamente al convertitore bidirezionale i caratteri da 0 a 255 attraverso la porta seriale (AUX=COM1), legge i caratteri che si trovano sull'ingresso parallelo dello stesso, e visualizza entrambi i valori sullo schermo fino al momento in cui si preme un tasto per terminare la routine.

Con il calcolatore e il convertitore non alimentati è possibile collegarli utilizzando il cavo opportu-

```
PROGRAMMA PER IL CONVERTITORE;
VAR
N:INTEGER;
V:CHAR;
BEGIN
REPEAT
N:=0;
REPEAT
WRITE(AUX,CHR(N));
READ(AUX,V);
WRITELN('DATO TRASMESSO: ',N:3,'—DATO RICEVUTO:',V:3);
DELAY(500);
N:=N+1;
UNTIL (N=256) OR (KEYDEPRESSED=TRUE);
UNTIL KEYDEPRESSED=TRUE
END.
```

namente realizzato; solo dopo questa operazione si può applicare l'alimentazione. Prima di eseguire il programma di verifica bisogna impostare con il comando DOS riportato di seguito la porta seriale con gli stessi parametri utilizzati per il convertitore: 8 bit per i dati, due bit di stop, nessuna parità e velocità di 1.200 baud:

JUMPER	CHIUSO	APERTO
J6	senza RTS e CTS	con RTS e CTS
J5	senza bit di parità	con bit di parità
J4	due bit di stop	un bit di stop
J3/J2	Lunghezza della parola	
J1	parità pari	parità dispari

J3 J2	LUNGHEZZA
0 0	5 bit
0 1	6 bit
1 0	7 bit
1 1	8 bit

La posizione dei ponticelli determina il formato del dato seriale e di quello parallelo. La programmazione deve coincidere con quella impostata nel calcolatore con il comando MODE

MODE COM1: 1200,N,8,2 <CR>

Dopo aver impostato la porta si può eseguire il programma, controllando i risultati che compaiono sullo schermo.

APPLICAZIONI

Molte sono le applicazioni alle quali può essere destinato questo convertitore bidirezionale, poiché può essere utilizzato come interfaccia universale per qualsiasi dispositivo controllato esternamente; è possibile ad esempio scambiare informazioni con una scheda di acquisizione dati progettata con convertitori Analogici/Digitali e Digitali/Analogici, oppure controllare dispositivi di potenza attraverso dei relé. Qualsiasi sia l'applicazione, chi deve gestire l'informazione è il programma di controllo, che deve essere elaborato personalmente dal lettore in funzione delle sue necessità. Il programma di prova che è stato utilizzato può servire come base per lo sviluppo delle proprie applicazioni. Il software deve comunque avere la seguente struttura:

1. *Configurazione della porta seriale.* Il formato dei dati selezionato sul convertitore deve coincidere con quello della porta seriale, tenendo sempre presente che la velocità è fissa a 1.200 baud.

2. *Gestione dell'informazione.* I dati ricevuti dal convertitore vengono analizzati in questa sezione; di conseguenza, il programma deve eseguire le azioni richieste dal processo di controllo preparando il dato che deve essere inviato attraverso la porta seriale.

3. *Presentazione dei risultati.* Deve essere visualizzato sullo schermo il processo di controllo, con le operazioni eseguite in funzione del dato ricevuto.
L'uscita per la stampante, se utilizzata, deve essere contemplata in questa sezione.

4. *Trasmissione dei dati.* I dati generati dall'elaborazione dell'informazione ricevuta devono essere trasmessi attraverso la porta seriale.

5. *Ricezione dei dati.* L'informazione ricevuta dal convertitore deve essere memorizzata in una variabile per una sua analisi successiva.

6. *Gestione degli errori.* Sempre che risulti possibile, si devono rilevare i probabili errori nello scambio dell'informazione, in modo da essere certi che il processo stia procedendo correttamente.

Elenco componenti

Resistenze
R1, R3, R12 = 100 kΩ
R2, R11 = 10 kΩ
R4, R5, R6, R10, R13 = 4,7 kΩ
R7, R14 = 22 kΩ
R8, R9 = 2,2 kΩ
R15 = 1 kΩ
R16 = 1 MΩ

Condensatori
C1, C2, C3 = 2,2 nF, ceramico
C4 = 3,9 nF, ceramico
C5 = 27 pF, ceramico
C6, C7, C8, C10, C12 = 100 nF, ceramico
C9 = 10 μF/16 V, elettrolitico
C11 = 1 μF/16 V, elettrolitico

Semiconduttori
D1, D2 = 1N4148
T1, T2, T3, T4 = BC547
IC1 = UART 8502 o AY-5-1013 o CPD1854 o 8725 ecc.
IC2, IC3 = 74HC241
IC4 = 74HC4060
IC5 = 74HC132
X1 = quarzo da 2,4576 MHz

Varie
Circuito stampato PC10190V0647
39 Terminali per c.s. maschi
6 Ponticelli di programmazione
22 Terminali femmina
110 Terminali torniti per zoccoli
1 Connettore tipo D a 25 o 9 terminali
Cavo per trasmissione dati con almeno cinque conduttori

Circuito completamente montato